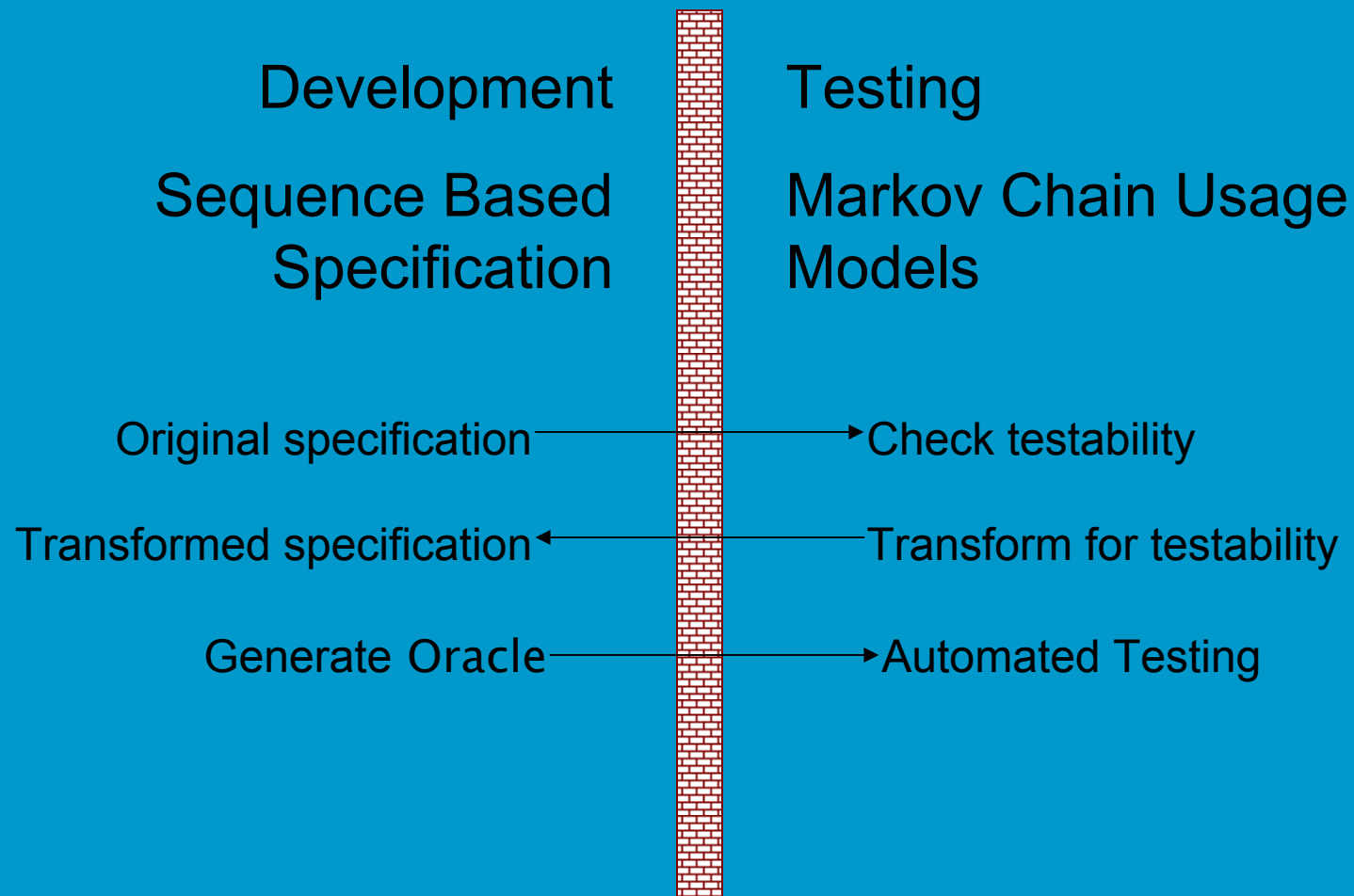


# Design for Testability

Department of Computer Science  
The University of Tennessee  
<http://www.cs.utk.edu/sqrl/>

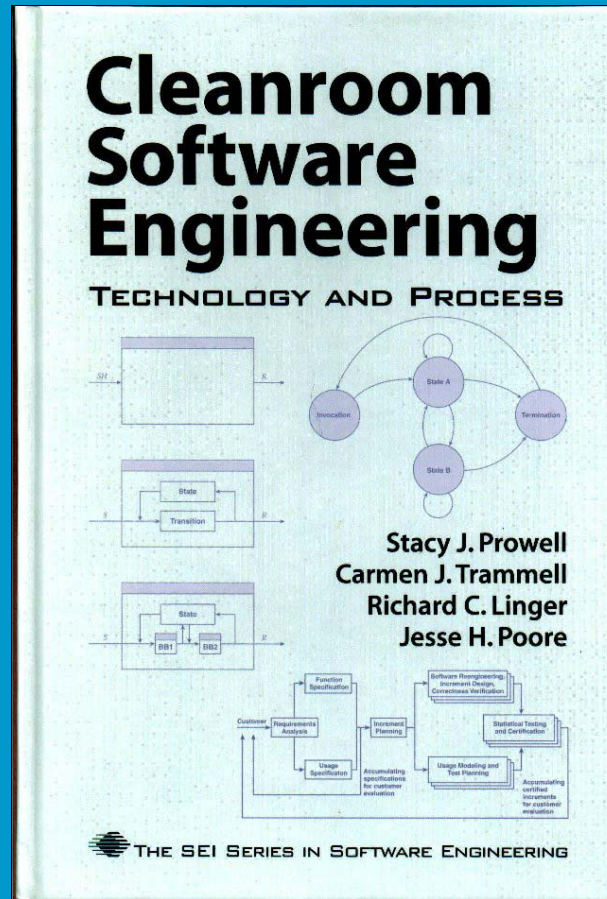
# Design for Testability



# Issues

- Requirements for testability
- Efficient, automated testing
- Automated oracle

# *Cleanroom Software Engineering: Technology and Process*

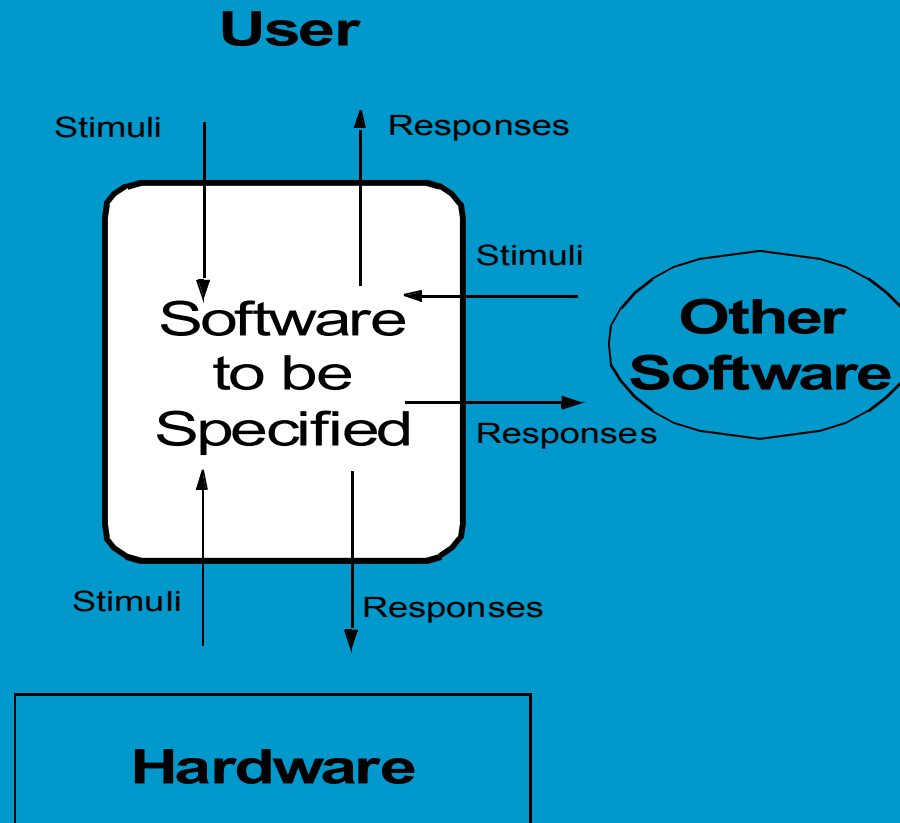


Addison Wesley  
ISBN 0-201-85480-5

# Sequence Based Specification

1. Define System Boundary
2. Enumerate Stimulus Sequences
3. Analyze Canonical Sequences
4. Define Specification Functions
5. Construct Table of Rules

# System Boundary and Interfaces



# The Security Alarm Example

p. 46 in *Cleanroom Software Engineering*

Current Stimulus: _____			
Current State	Response	State Update	Black Box Trace: Sequence Prior to Current Stimulus

Figure 3.3 State box mapping table format

state box can be generated automatically from the black box, and need not be verified if generated by a certified tool.

The final form of the state box is a set of mapping tables, one per stimulus. Each mapping table is of the form shown in Figure 3.3.

The state box specification is the final specification work product. The Cleanroom box structure specification and design method continues with refinement of the state box to the clear box, in terms of full procedural design, as described in Chapter 4.

### 3.3 Example: Specification of a Security Alarm

A simple software-controlled security alarm depicted in Figure 3.4 is to be created for use on doors, windows, boxes, and so forth, to detect unauthorized entry. The security alarm has a detector that sends a trip signal when motion is detected. The security alarm is activated by pressing the Set button. A light in the Set button is illuminated when the security alarm is on. If a trip signal occurs

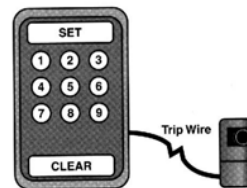


Figure 3.4 Security alarm

while the device is set, a high-pitched tone (alarm) is emitted. A three-digit code must be entered to turn off the alarm. Correct entry of the code deactivates the security alarm. If a mistake is made when entering the code, the user must press the Clear button before the code can be reentered. The security alarm will not be programmable; each unit will have a hard-coded deactivation code.

A sequence-based specification will be created for the security alarm using the stepwise process described in the preceding subsection.

#### 3.3.1 Black Box Definition

**Tagged Requirements.** Tagging of requirements is the first step in creating a traceable specification, as shown in Table 3.3. Subsequent elements of the specification will be traced to their origin in the requirements through these tags.

As each step in the specification is traced to the relevant requirement, ambiguities and omissions in the requirements will be discovered. When there is no requirement to cite in a trace, a "derived" requirement will be stated and tagged as D1, D2, and so on.

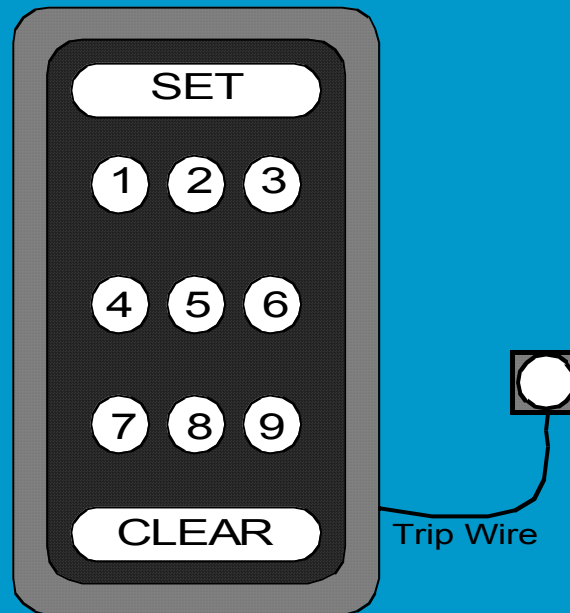
**System Boundary Definition.** There are two possible sources of stimuli to the security alarm: the detector and the human user. The detector sends a trip stimulus and all other stimuli originate with the human user, as shown in Table 3.4.

The stimuli Trip, Set, and Clear are all atomic stimuli (i.e., discrete, low-level stimuli). The stimuli GoodDigit and BadDigit are both abstractions, representing correct and incorrect entry of digits in the three-digit code. GoodDigit represents each digit in the sequence of three digit entries that deactivate the device. BadDigit represents a digit in any other sequence of digit entries.

Table 3.3 Tagged requirements for the security alarm

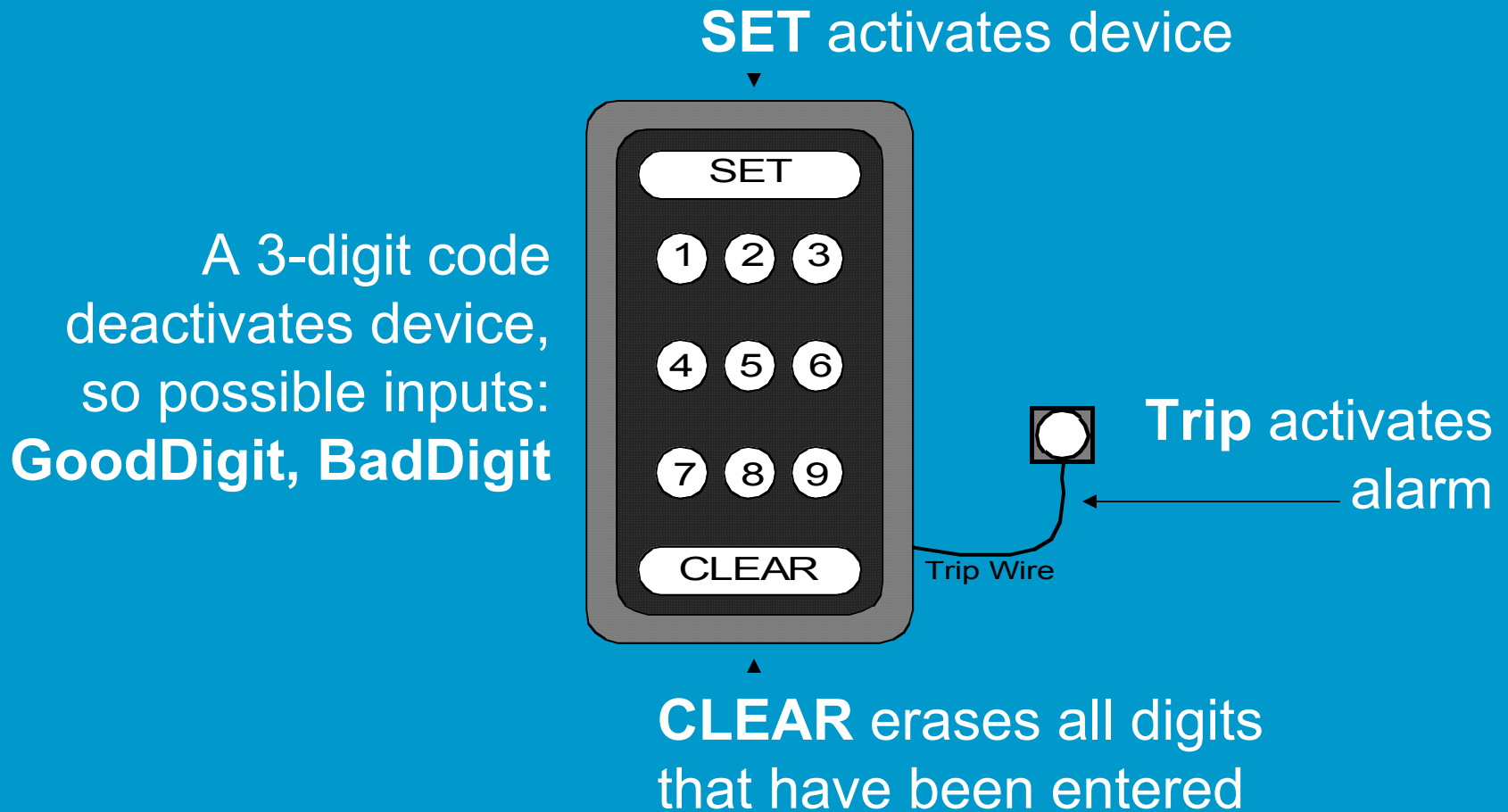
Tag No.	Requirement
1	The security alarm has a detector that sends a trip signal when motion is detected.
2	The security alarm is activated by pressing the Set button.
3	The Set button is illuminated when the security alarm is set.
4	If a trip signal occurs while the security alarm is set, a high-pitched tone (alarm) is emitted.
5	A three-digit code must be entered to turn off the alarm tone.
6	Correct entry of the code deactivates the security alarm.
7	If a mistake is made when entering the code, the user must press the Clear button before the code can be reentered.

# The Security Alarm





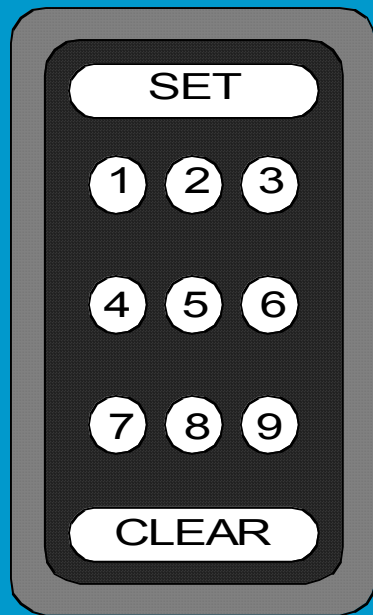
# Security Alarm Stimuli



# Stimuli - Symbols

Stimulus	Description	Symbol
Set	Device activator	S
Trip	Signal from detector	T
GoodDigit	A digit that is part of the correct entry of the three-digit code that deactivates the alarm and device.	G
BadDigit	Incorrect entry of a digit in the code	B
Clear	Clear entry	C

# Security Alarm Responses



SET button

- **Light on** when device is activated
- **Light off** when correct code entered



Alarm siren

- **Alarm on** when tripped
- **Alarm off** when correct code entered

# Responses - Symbols

Response	Description	Symbol
Light On	Set button illuminated	L-on
Light off	Set button not illuminated	L-off
Alarm On	Alarm tone activated	A-on
Alarm Off	Alarm tone deactivated	A-off

# Requirements

- Individual requirements tagged for traceability.
- Initial requirements assumed to be incomplete, inconsistent, and possibly incorrect.
- Enumeration is a systematic process for discovering omissions and ambiguities.
- Domain experts review clarified requirements for correctness.

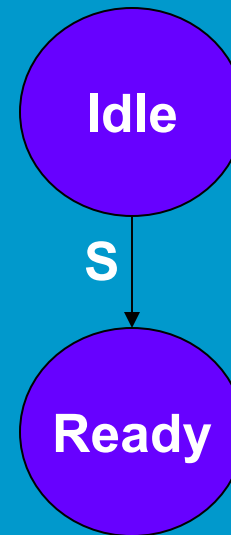
# Security Alarm Requirements

Tag	Requirement
1	The security alarm has a detector that sends a trip signal when motion is detected.
2	The security alarm is activated by pressing the SET button.
3	The SET button is illuminated when the security alarm is set.
4	If a trip signal occurs while the security alarm is set, a tone (alarm) is emitted.
5	A three-digit code must be entered to silence the alarm tone.
6	Correct entry of the code deactivates the security alarm.
7	If a mistake is made when entering the code, the user must press the CLEAR button before the code can be reentered.

# Scenarios of Use (length 1)

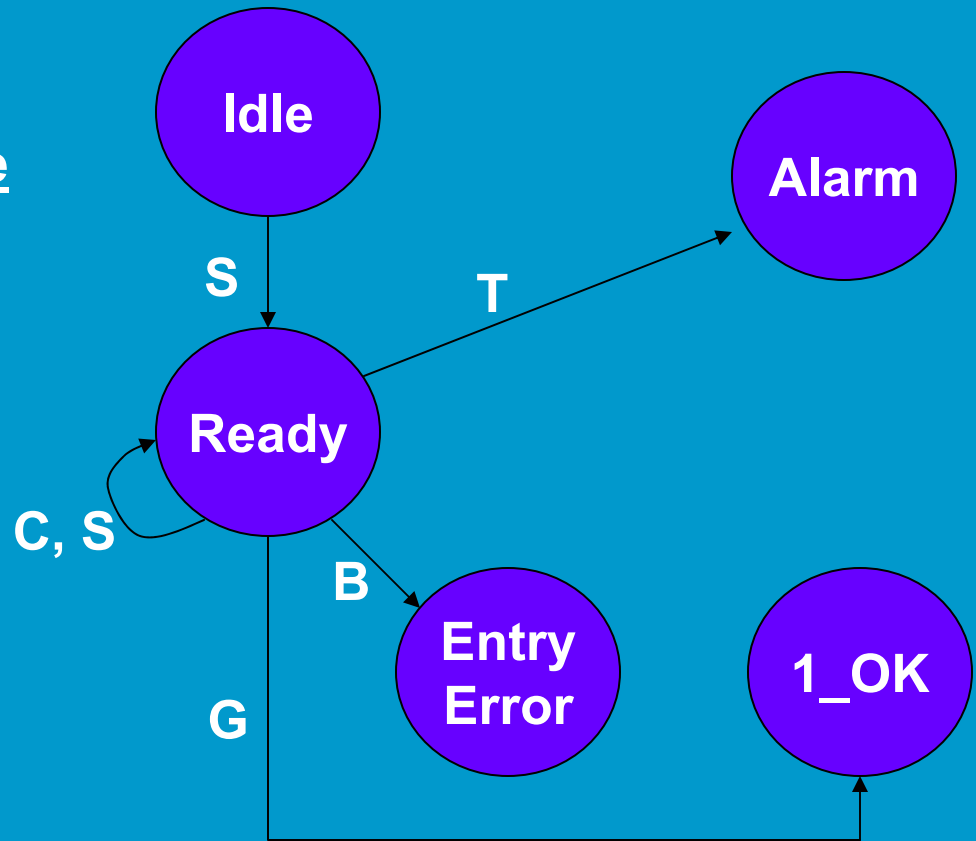
The device is initially Idle.

<u>Scenario</u>	<u>Status of Device</u>
<b>S</b>	<b>L-on; Ready</b>
<i>T</i>	<i>no change</i>
<i>G</i>	<i>no change</i>
<i>B</i>	<i>no change</i>
<i>C</i>	<i>no change</i>



# Scenarios of Use (length 2)

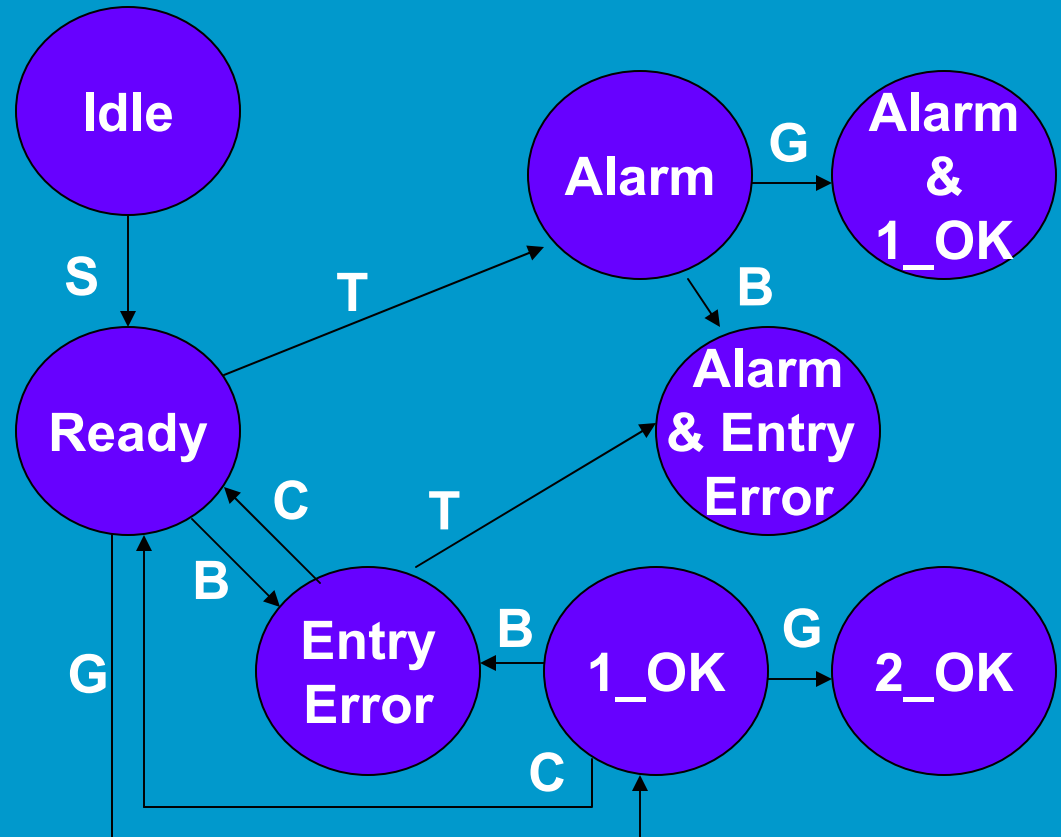
<u>Scenario</u>	<u>Status of Device</u>
SS	<i>no change</i>
ST	<b>A-on; Alarm</b>
SG	<b>1_OK</b>
SB	<b>Entry Error</b>
SC	<i>no change</i>





# Scenarios of Use (length 3)

Stimuli	Status of Device
STS	<i>no change</i>
STT	<i>no change</i>
<b>STB</b>	<b>Alarm &amp; Entry Error</b>
STC	<i>no change</i>
<b>STG</b>	<b>Alarm &amp; 1_OK</b>
SBS	<i>no change</i>
<b>SBT</b>	<b>A-on; Alarm</b>
SBB	<i>no change</i>
<b>SBC</b>	<b>Ready</b>
SBG	<i>no change</i>
SGS	<i>no change</i>
<b>SGT</b>	<b>A-on; ?</b>
<b>SGB</b>	<b>Entry Error</b>
<b>SGC</b>	<b>Ready</b>
<b>SGG</b>	<b>2_OK</b>



# The Specification is Clarified

Stimuli	Status of Device
STS	<i>no change</i>
STT	<i>no change</i>
STB	Alarm & Entry Error
STC	<i>no change</i>
STG	Alarm & 1_OK
SBS	<i>no change</i>
SBT	<b>A-on</b> ; Alarm
SBB	<i>no change</i>
SBC	Ready
SBG	<i>no change</i>
SGS	<i>no change</i>
<b>SGT</b>	<b>A-on</b> ; ?
SGB	Entry Error
SGC	Ready
SGG	2_OK

Does the Good Digit count as part of the correct entry?

**DECISION: No. All three digits must be entered after the alarm.**

# Security Alarm Enumeration

Sequence	Response	Equivalence	Trace
$\lambda$ (empty)	null		D1
S	Light On		2,3
all others	illegal		D1
SS	null	S	D2
ST	Alarm On		4
SB	null		D3
SC	null	S	D4
SG	null		D5

# Enumeration: Sequences of Length 3

Sequence	Response	Equivalence	Trace
STS	null	ST	D2
STT	null	ST	D6
STB	null		D3
STC	null	ST	D4
STG	null		D5
SBS	null	SB	D2
SBT	Alarm On	STB	4
SBB	null	SB	D3
SBC	null	S	D4,7
SBG	illegal		7
SGS	null	SG	D2
SGT	Alarm On	STB	4,D7
SGB	null	SB	D3
SGC	null	S	D4
SGG	null		D5

# Enumeration: Sequences of Length 4

Sequence	Response	Equivalence	Trace
STBS	null	STB	D2
STBT	null	STB	D6
STBB	null	STB	D3
STBC	null	ST	D4,7
STBG	illegal		7
STGS	null	STG	D2
STGT	null	STG	D6
STGB	null	STB	D3
STGC	null	ST	D4
STGG	null		D5
SGGS	null	SGG	D2
SGGT	Alarm On	STB	4,D7
SGGB	null	SB	D3
SGGC	null	S	D4
SGGG	Light Off	I	6

# Enumeration: Sequences of Length 5

Sequence	Response	Equivalence	Trace
STGGS	null	STGG	D2
STGGT	null	STGG	D6
STGGB	null	STB	D3
STGGC	null	ST	D4
STGGG	Alarm Off Light Off	$\lambda$	3,5,6

# Development Continues

- Complete the Specification
  - Complete
  - Consistent
  - Traceably correct
- Derive State Machine(s)
- Design
- Code
- Verify

# Testability



Testing

Markov Chain Usage  
Models



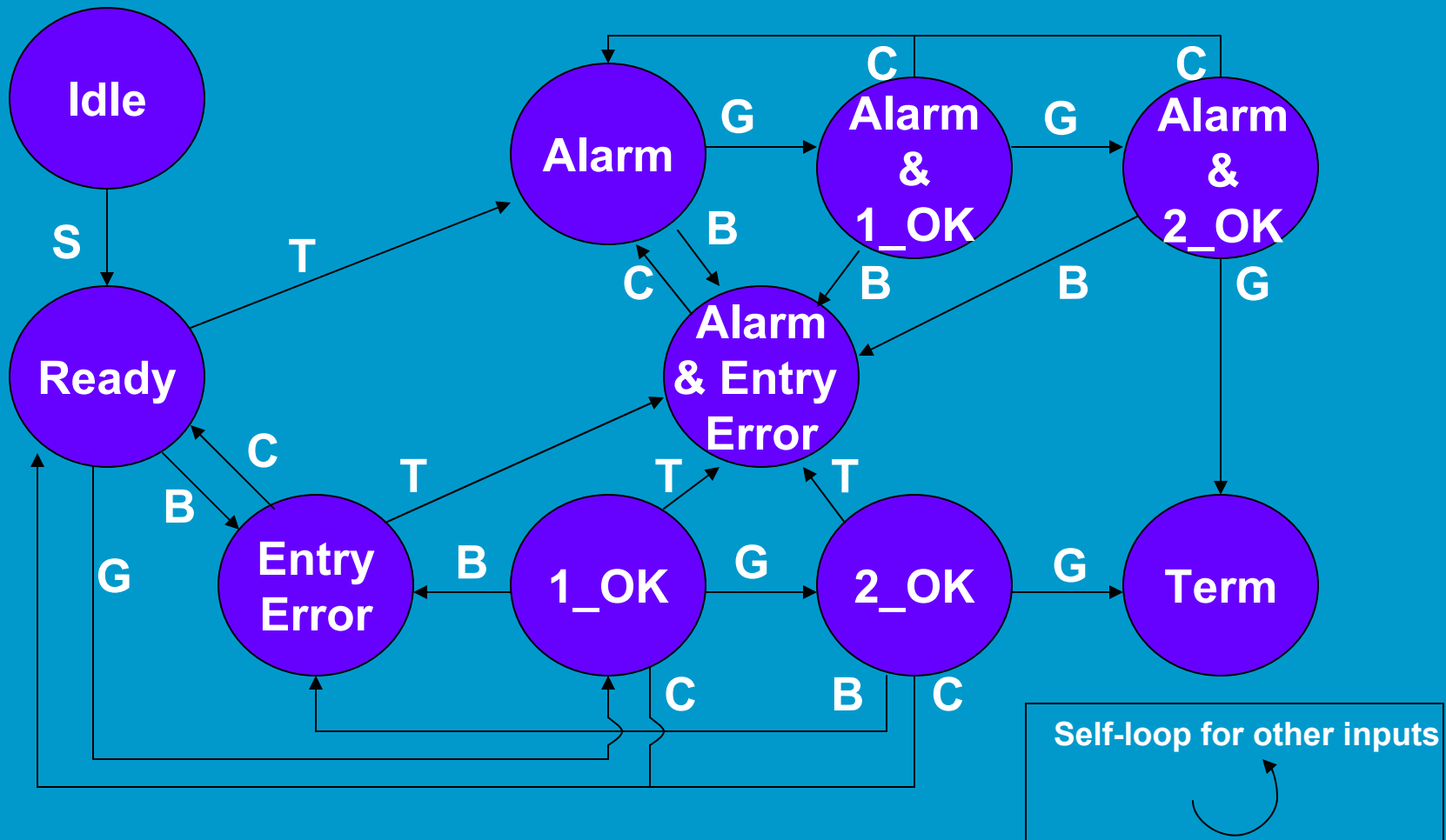
# Markov Chain Usage Model Testing

1. Define Certification Plan
2. Build Usage Model
3. Determine State Transition Probabilities
4. Validate Usage Model
5. Generate Test Cases, Execute, Evaluate

# Security Alarm Test Plan

- Perform a successful coverage test.
- Perform 60 consecutive successful random tests that sound the alarm.
- Perform 60 consecutive successful random tests that do not sound the alarm.
- Perform unbiased random testing until 460 consecutive test cases have been run successfully.
- Calculate system reliability.

# Security Alarm Model Structure



# Model Structure enables Coverage Test

**Graph traversed via *Chinese Postman* algorithm**

Idle **S** Ready **G** 1\_OK **S** 1\_OK **G** 2\_OK **S** 2\_OK **C** Ready **G** 1\_OK **C** Ready **G** 1\_OK **G** 2\_OK **B** EntryError  
**C** Ready **S** Ready **C** Ready **T** Alarm **G** Alarm&1\_OK **S** Alarm&1\_OK **T** Alarm&1\_OK **C** Alarm **G**  
Alarm&1\_OK **G** Alarm&2\_OK **S** Alarm&2\_OK **T** Alarm&2\_OK **B** Alarm&EntryError **C** Alarm **G**  
Alarm&1\_OK **G** Alarm&2\_OK **C** Alarm **S** Alarm **C** Alarm **T** Alarm **G** Alarm&1\_OK **B** Alarm&EntryError  
**C** Alarm **G** Alarm&1\_OK **G** Alarm&2\_OK **G** Term

Idle **S** Ready **G** 1\_OK **G** 2\_OK **G** Term

Idle **S** Ready **G** 1\_OK **G** 2\_OK **T** Alarm&EntryError **C** Alarm **G** Alarm&1\_OK **G** Alarm&2\_OK **G** Term

Idle **S** Ready **G** 1\_OK **T** Alarm&EntryError **C** Alarm **G** Alarm&1\_OK **G** Alarm&2\_OK **G** Term

Idle **S** Ready **G** 1\_OK **B** EntryError **S** EntryError **B** EntryError **G** EntryError **C** Ready **B** EntryError **T**  
Alarm&EntryError **S** Alarm&EntryError **B** Alarm&EntryError **T** Alarm&EntryError **G** Alarm&EntryError **C**  
Alarm **B** Alarm&EntryError **C** Alarm **G** Alarm&1\_OK **G** Alarm&2\_OK **G** Term

**Minimum number of steps to cover all states and arcs**

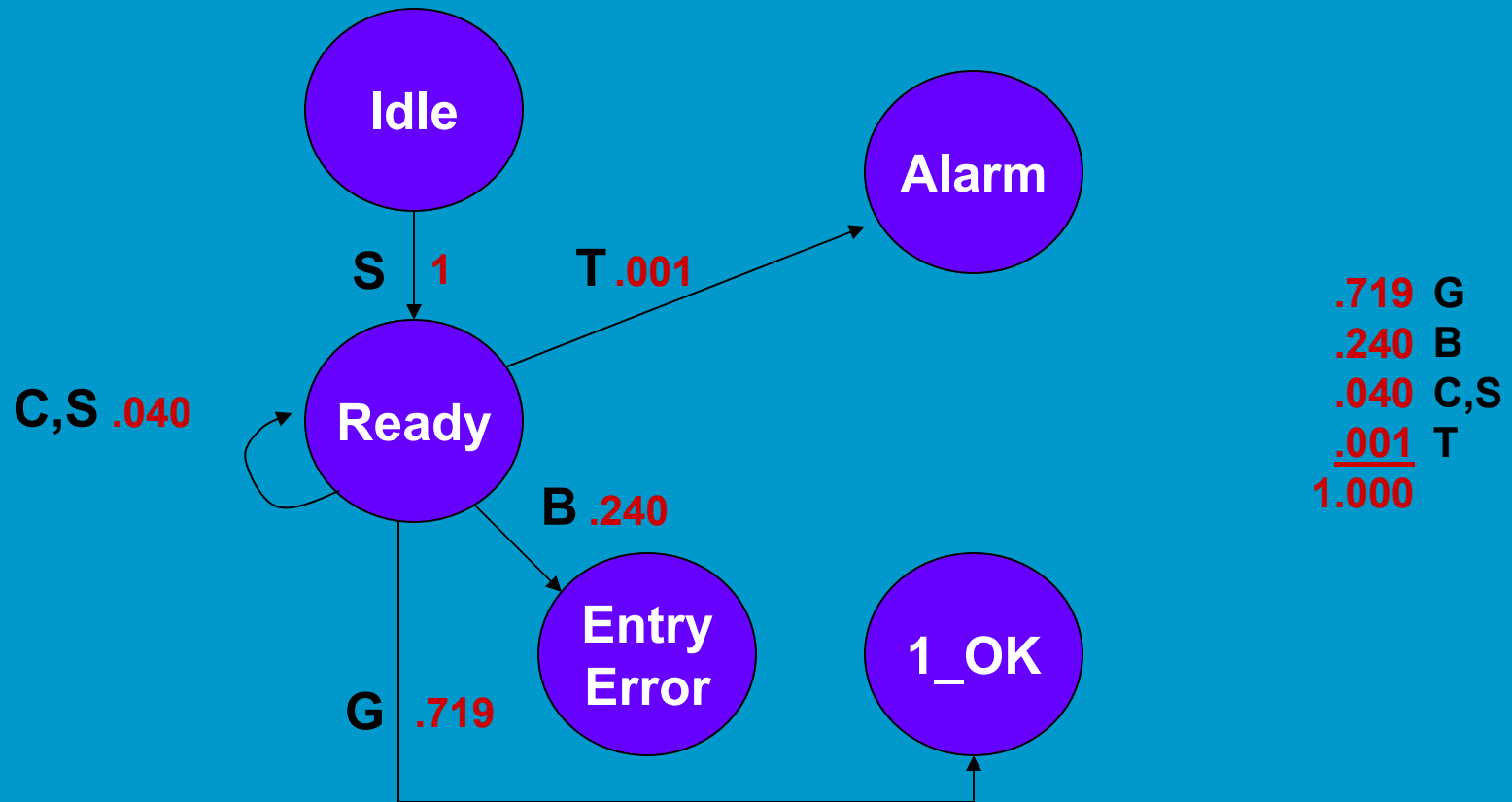
# Graph: All Possible Use

## Arc Probabilities: Expected Use

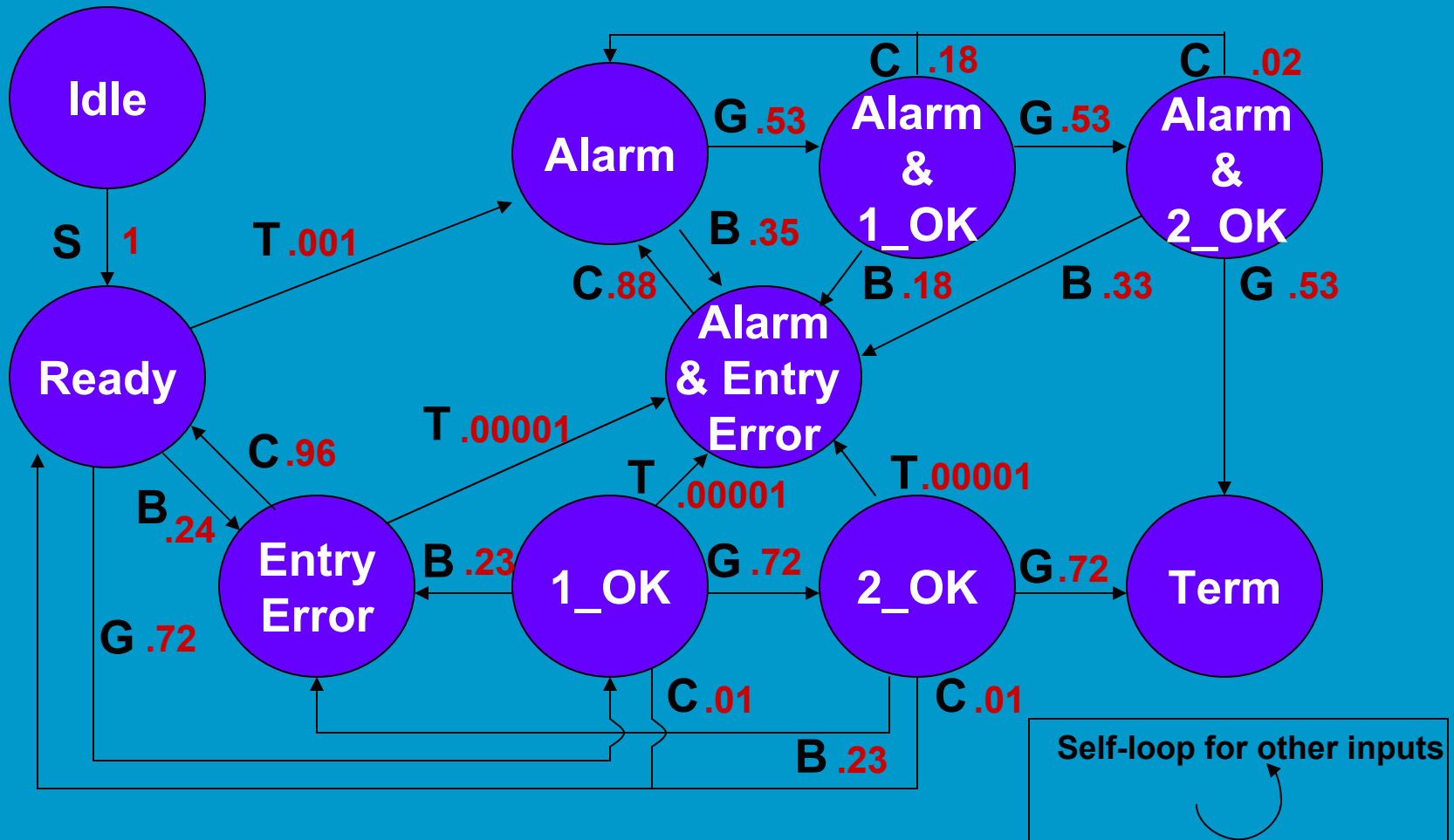
- The structure of the usage model (the graph) represents all possible uses of the software.
- Transition probabilities on the arcs represents the expected use.

# Transition Probabilities on *Exit Arcs* sum to 1

Probabilities come from historical data or engineering judgment



# Security Alarm Usage Model



# Graph represented as Matrix enables *Markov Analysis*

TO

FROM	Idle	Ready	Entry Error	1_OK	2_OK	Alarm	Alarm &1OK	Alarm &2OK	Alarm &EE	Term
Idle	0	1	0	0	0	0	0	0	0	0
Ready	0	.0400	.2398	.7193	0	.0010	0	0	0	0
EntryError	0	.9590	.0400	0	0	0	0	0	.00001	0
1_OK	0	.0096	.2304	.0400	.7200	0	0	0	.00001	0
2_OK	0	.0096	.2304	0	.0400	0	0	0	.0001	.7200
Alarm	0	0	0	0	0	.1200	.5280	0	.3520	0
Alarm&1OK	0	0	0	0	0	.1760	.1200	.5280	.1760	0
Alarm&2OK	0	0	0	0	0	.0176	0	.1200	.3344	.5280
Alarm&EE	0	0	0	0	0	.8800	0	0	.1200	0
Term	1	0	0	0	0	0	0	0	0	0



# Standard Analysis yields *Operational Profile*

Expected long-run occupancy for each

- state
- arc
- stimulus

# Operational Profile for the Security Alarm

State	Long run occupancy
Idle	.109597
Ready	.270206
Entry Error	.152519
1_OK	.202452
2_OK	.151837
Alarm	.001447
Alarm & 1_OK	.000868
Alarm & 2_OK	.000521
Alarm & Entry Error	.000956
Term	.109597

← Most likely state

← Least likely state

**These results are validated against data and expert opinion.**

# Operational Profile for the Security Alarm

FROM STATE	Arc	Long run occupancy
Idle	Set	.12308685
	BadDigit	.07275884
Ready	GoodDigit	.21827653
	Set, Clear	.01212647
	Trip	.00030316
	Clear	.16443191
Entry Error	Set, BadDigit, GoodDigit	.00685811
	Trip	.00000171
	Trip	.00000170

Most likely arc

Least likely arc

These results are validated against data and expert opinion.

# Operational Profile for the Security Alarm

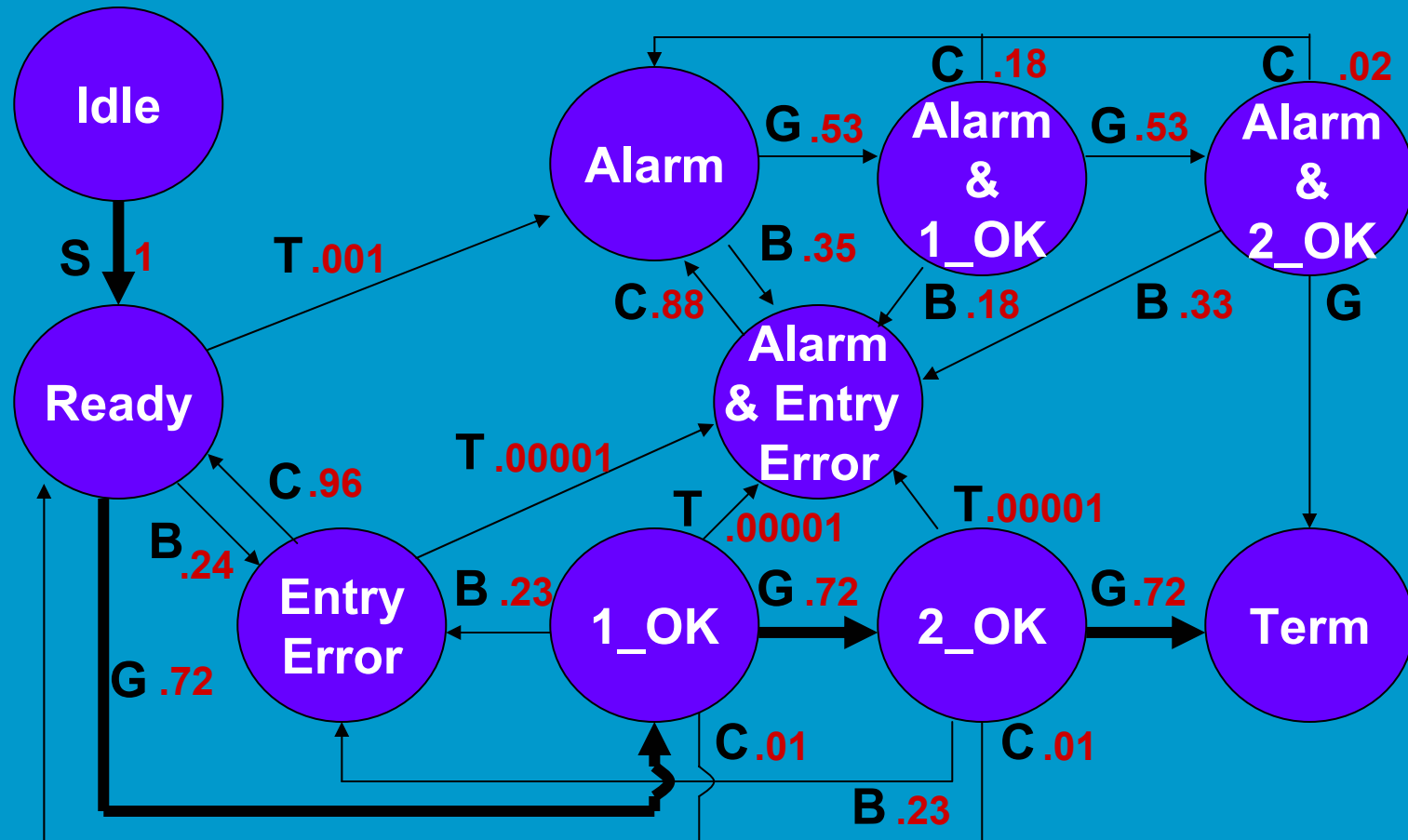
Stimulus	Long run occupancy
Set	0.13900261
GoodDigit	0.50644176
BadDigit	0.16537277
Clear	0.16937841
Trip	0.00030885
All self-loops	0.0195

← Most likely stimulus

← Least likely stimulus

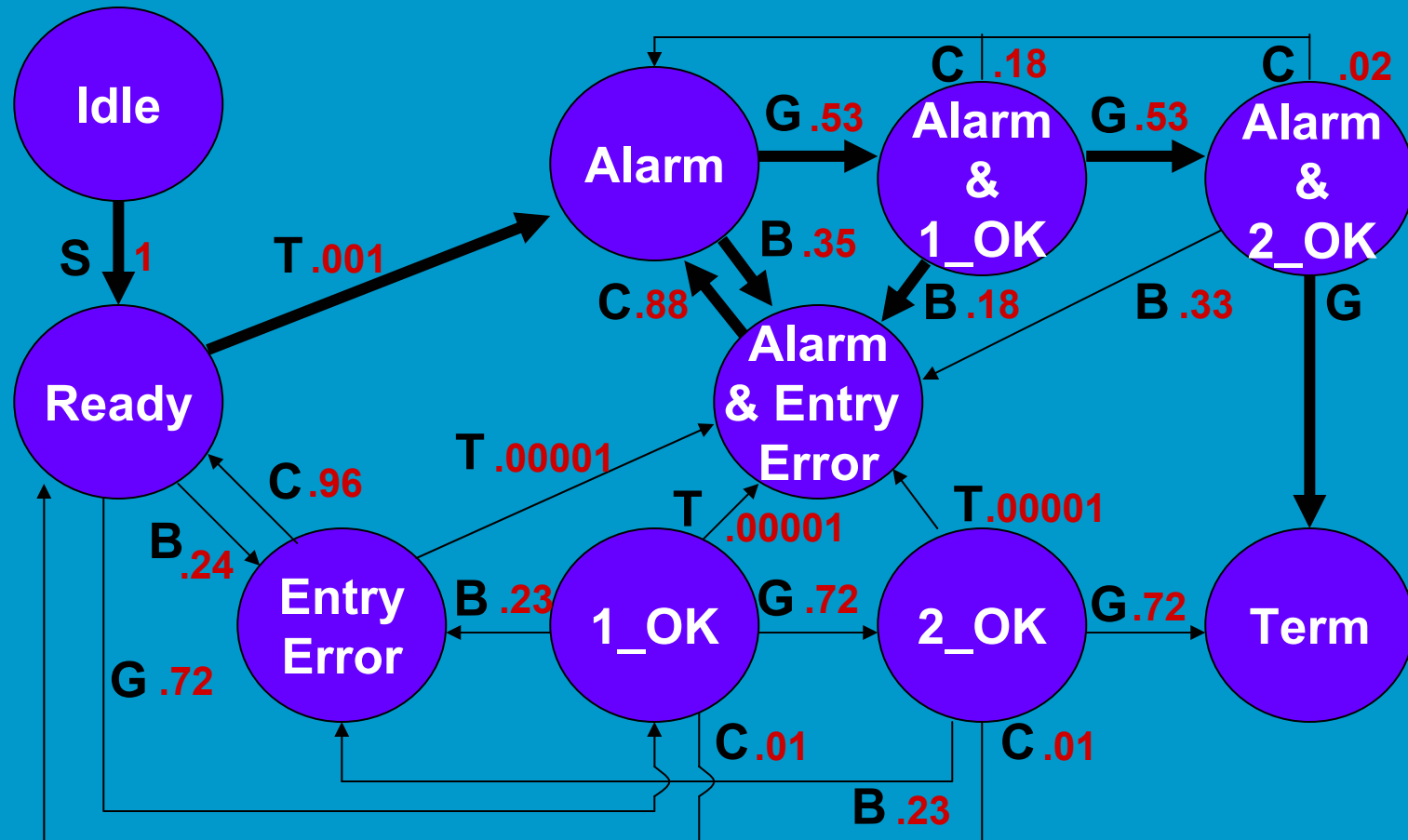
**These results are validated against data and expert opinion.**

# Importance Tests cover arcs based on “value”



**In this example, the most likely arcs are traversed.**

# Random Tests are *paths* based on random number




# Security Alarm Test Data

- Coverage test
  - 4 failures in 5 test cases found and fixed
  - coverage test successfully rerun
- Random tests that sound alarm (tests including T)
  - 4 failures in 15 tests found and fixed
  - 60 more tests run successfully
- Random tests that do not sound alarm (tests w/o T)
  - 2 failures in 15 random tests found and fixed
  - 60 more tests run successfully
- 460 more random test cases run successfully

# Test Analysis

## *Chronology of Test Activity*

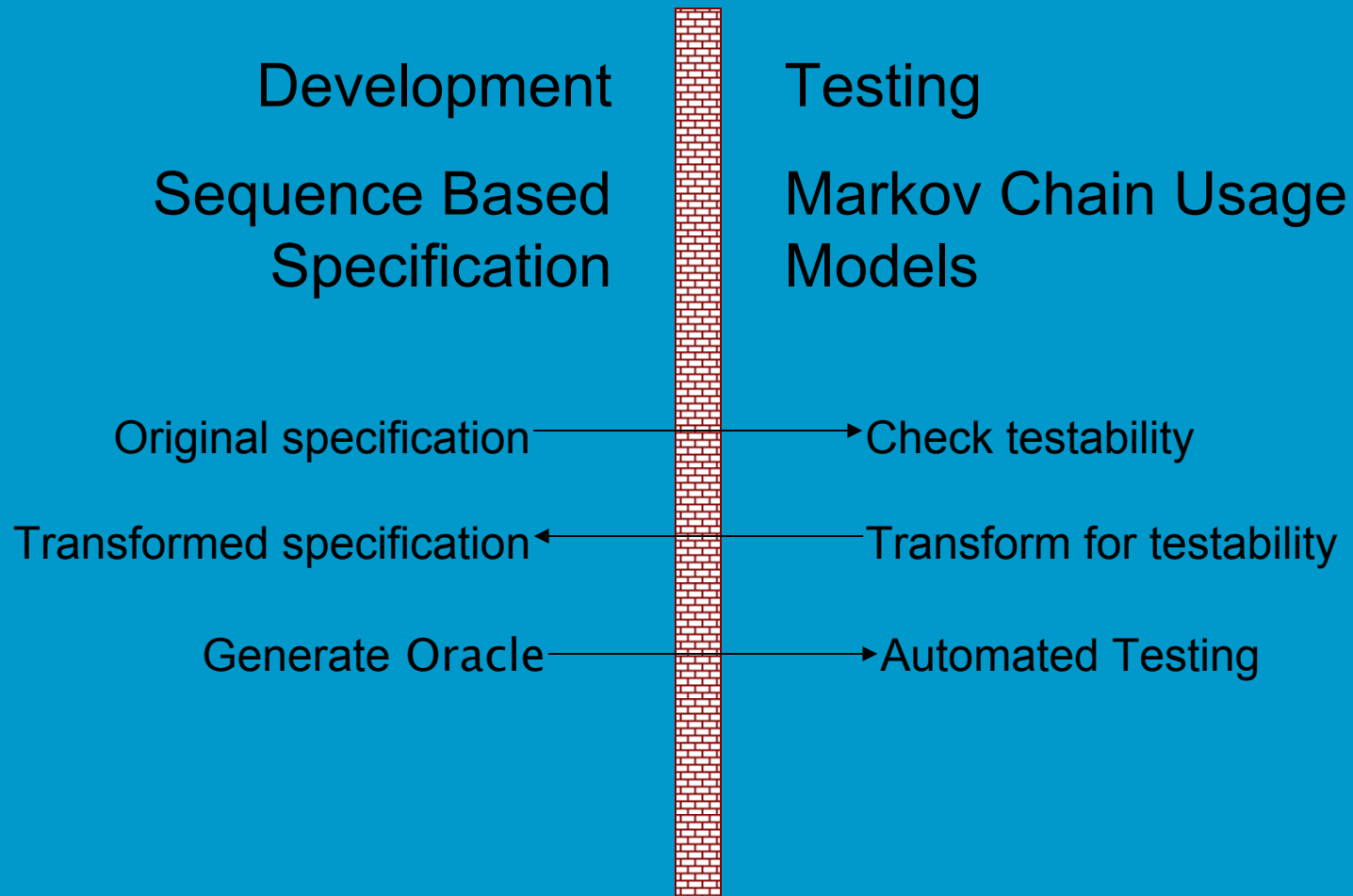


	<u>Arc Coverage</u>	<u>Alarm On</u>	<u>Alarm Off</u>	<u>Unbiased</u>
Test Cases	5	80	155	615
Test Events	65	1827	2910	6517
Event Reliability	0.77	0.96	0.9918	0.997
Use Reliability	0.22	0.72	0.94	0.97
Kullback	0.48	0.52	0.14	0.026

*(Cumulative statistics as testing proceeds)*



# Design for Testability



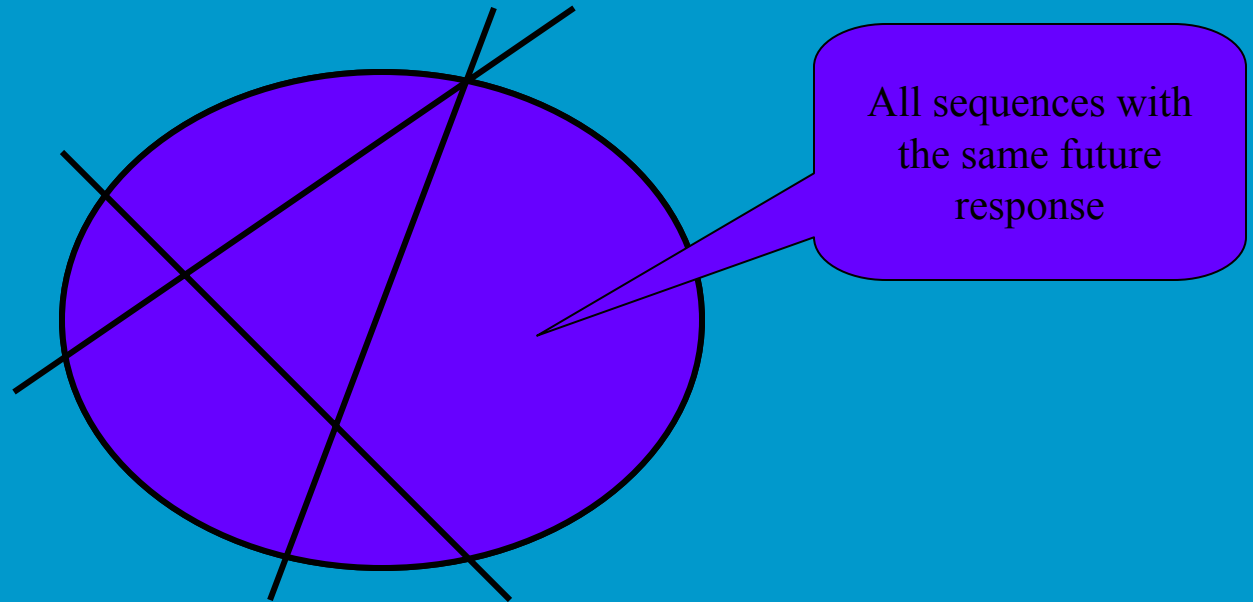
# Goal of Design for Testability

Fully automated, efficient testing

- Automatic generation of usage model from specification
- Automatic test case generation
- Automatic test case modification
- Automatic oracle

# Sequence-Based Specifications

- Every sequence
- State machine → usage model structure



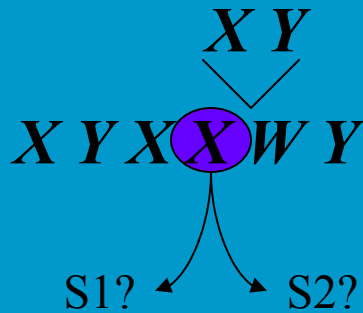
# Requirements for Testability

## Examination of sequence based specification

- Aspects that will be difficult to test: Derived requirements
- Ways to improve efficiency of testability

# Efficiency of Testability

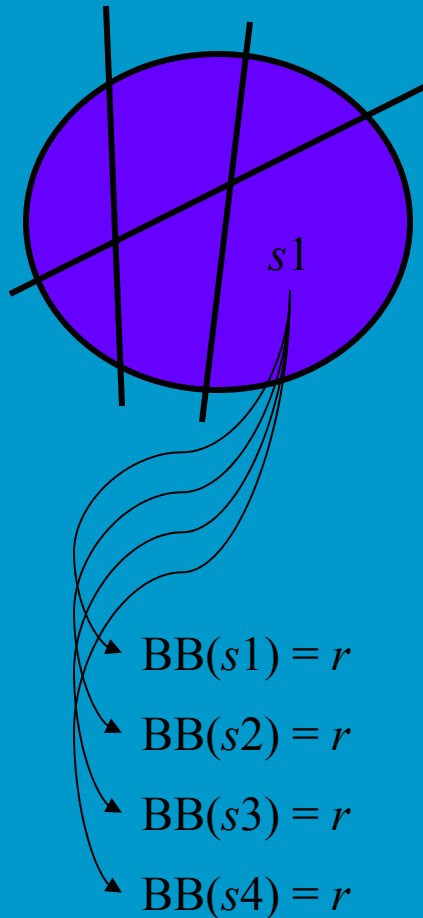
- Regular expressions for all possible outputs
  - Primes for testing (graph theory)



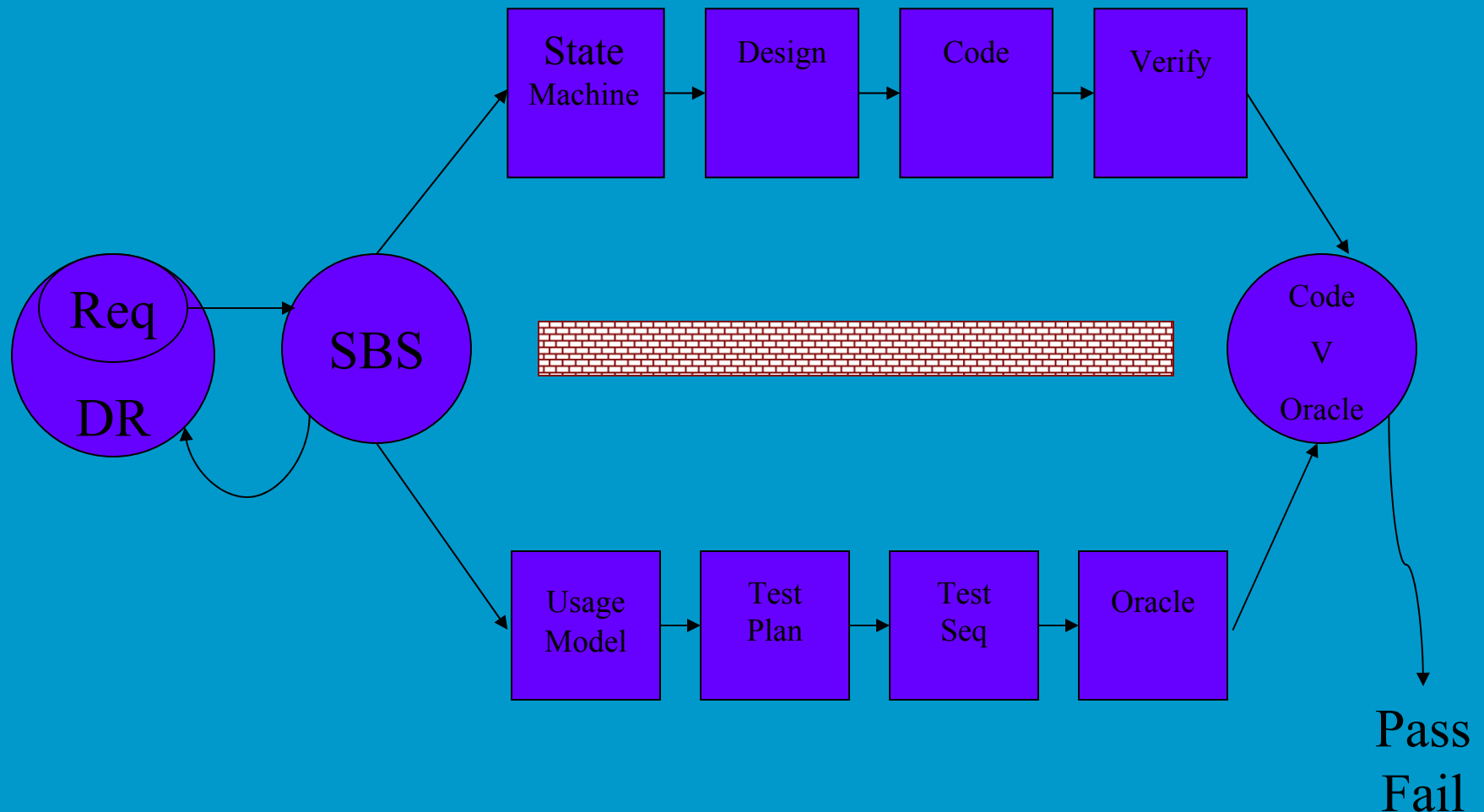
- Resolving ambiguity
  - Identify ambiguity
  - Sequence lookahead and know when (if) it will resolve
  - Splice in “helper” sequence
  - Special testability states

# Oracle

- $S^* / RE$
- Simulate execution of the specification  $\leftrightarrow$  execution of SUT
  - Given an executed sequence belonging to a block of a partition, sample other elements of the block using the known for an oracle
  - Within the block: 2 that give same response, 3 that give same response, ..., n that give same response



# Automated Testing Process



# Next Steps

- Publish research
- Tool support
- Get Raytheon to try it